Relatório 1º projecto ASA 2020/2021

Grupo: al052

Aluno(s): Francisco Oliveira (96860) e Martim Correia (97326)

Descrição do Problema e da Solução

O problema que se pretende resolver é dado um conjunto de dominós, representado por um DAG, qual a maior sequência de dominós derrubados, l, e quantas intervenções são necessárias para derrubar todos os dominós, k.

Para obter k, identificamos o tipo de cada nó (se é isolado, source, sink ou outro), sendo k o número de nós ou do tipo isolado ou do tipo source. Para obter l, encontramos uma ordem topológica com DFS, iteramos pela ordem inversa, observamos as adjacências e gravamos em cada nó a maior distância, sendo l a maior das distâncias.

Referências: https://www.geeksforgeeks.org/iterative-depth-first-traversal/

Análise Teórica

- Processar input:
  - Ler a primeira linha. **O(1)**
  - Inicializar grafo. **O(V)**
  - Ver arestas para identificar o tipo de cada nó e determinar k. k é inicialmente igual a V, usa-se um vetor que guarda o tipo de cada nó e decrementa-se k por 1 cada vez que um vértice deixar de ser isolado ou source. **O(E)**
  - Devolve-se o k. **O(1)**
  - Logo para esta etapa **O(V+E)**
- Obter ordem topológica invertida com a DFS. **O(V+E)**
  - DFS_iterative(G, s)

```
bool visited[G.V], finished[G.V];
stack = {};
stack.push(s);
while(stack != {})
    u  = stack.top();
    if (finished[u])
        stack.pop(); continue;
    if (!visited[u])
        visited[u] = true;
    // continua na próxima página
```

```
finished[u] = true;
for each v in adj[u]
        if (!visited[v])
                stack.push(v);
                if (finished[u])
                        finished[u] = false;
if (finished[v])
        stack.pop();
```

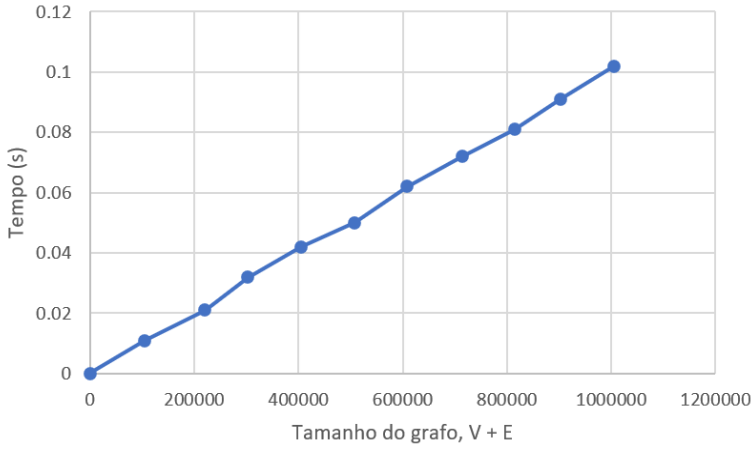
- Obter l
  - Percorrer a ordem topológica invertida. **O(V)**
  - Olhar para as adjacências e determinar a maior distância (este ciclo é chamado pelo menos uma vez). **O(E)**
  - Logo para esta etapa **O(V+E)**

Complexidade global da solução: **O(V+E)**

Avaliação Experimental dos Resultados

Geramos vários grafos de tamanho incremental com o randomDAG.cpp e corremos o programa com o comando time no terminal para registar o tempo de execução.

Especificações da máquina: CPU: i7-10750H@2.60GHz, GPU: RTX 2060, RAM: 16GB.



Assim, e como previsto, o tempo de execução cresce linearmente com o tamanho do grafo.